

10

FIGURE 1

962260-26002280

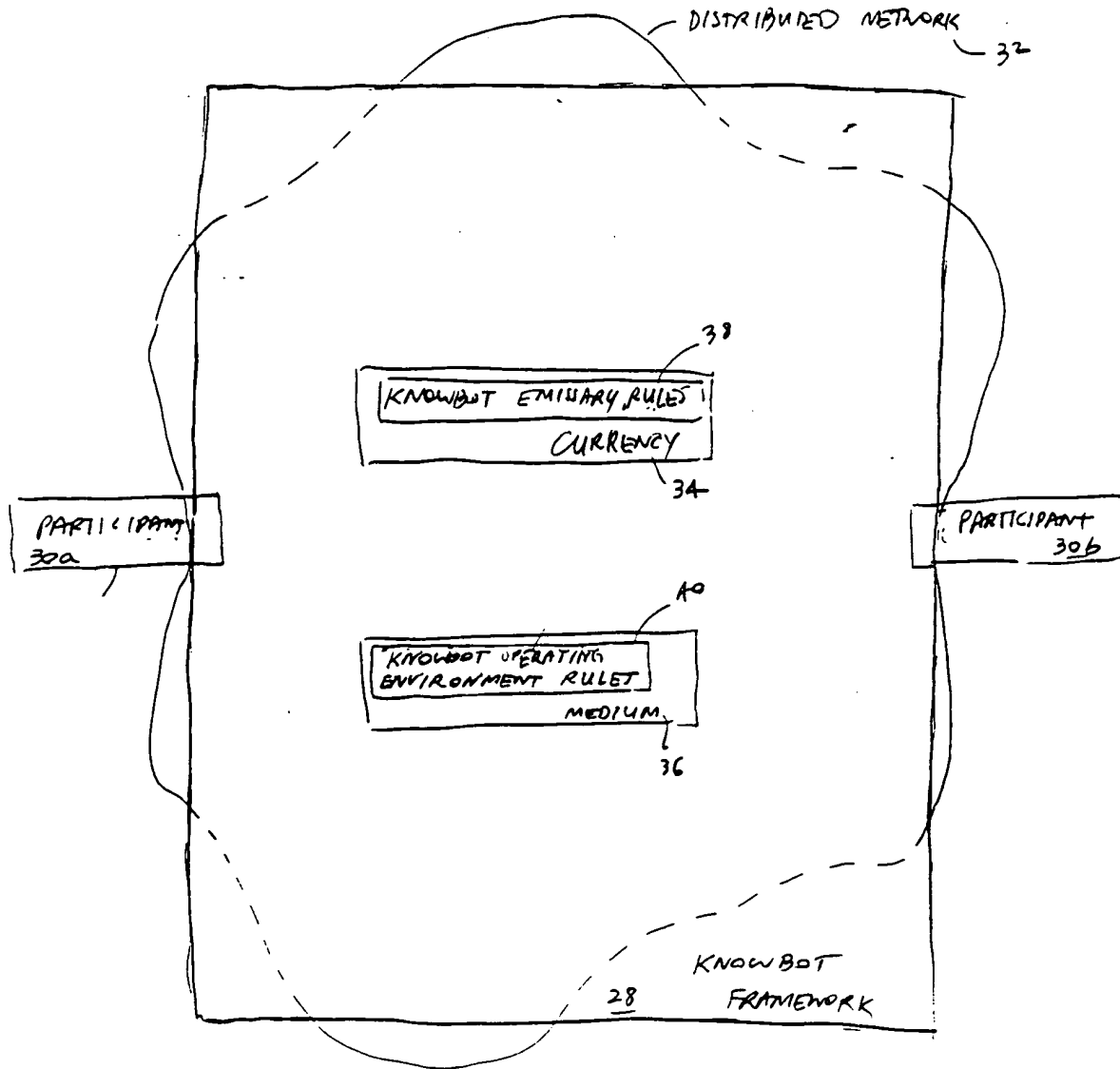


FIGURE 2

964260-26002480

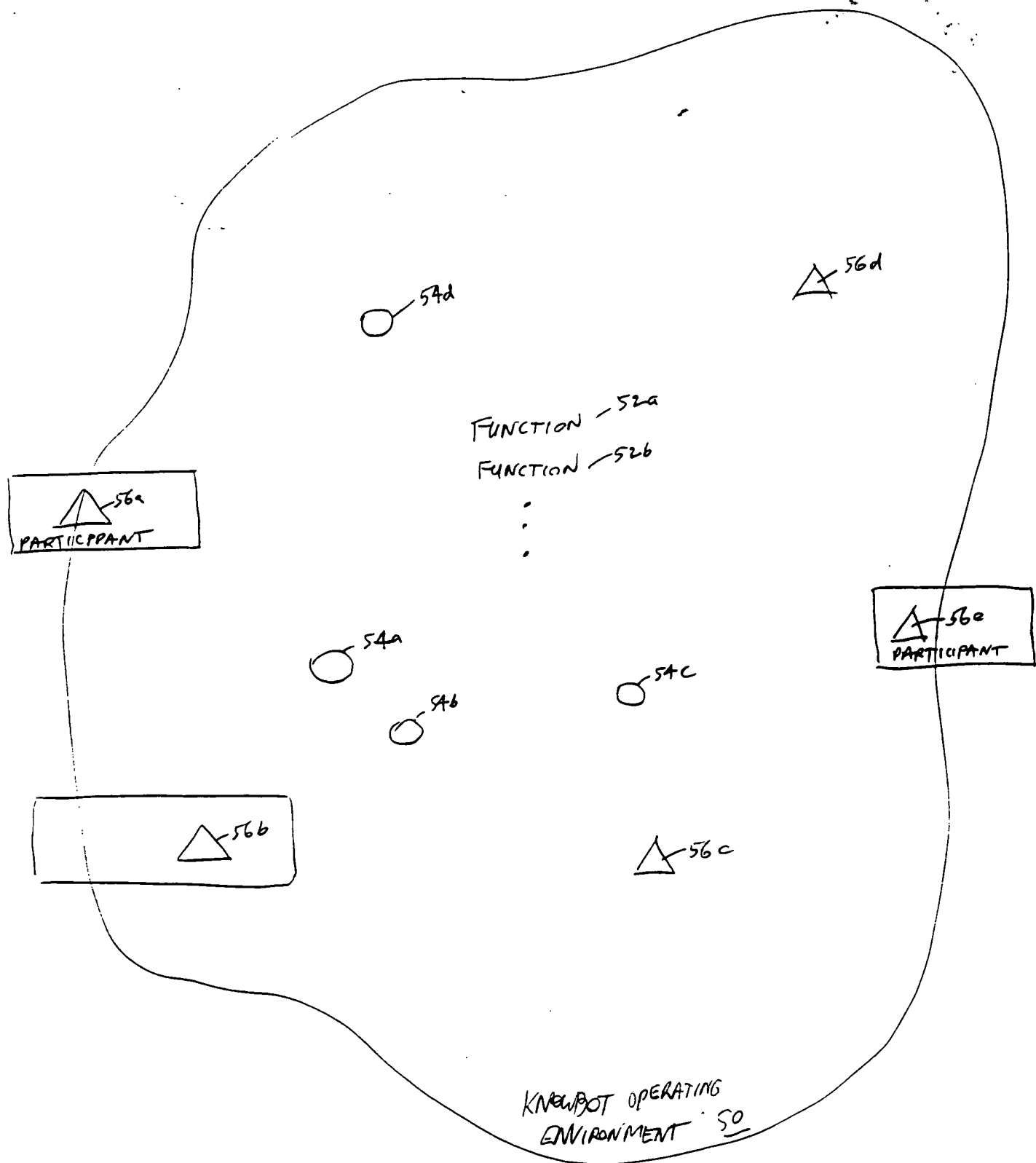


FIGURE 3

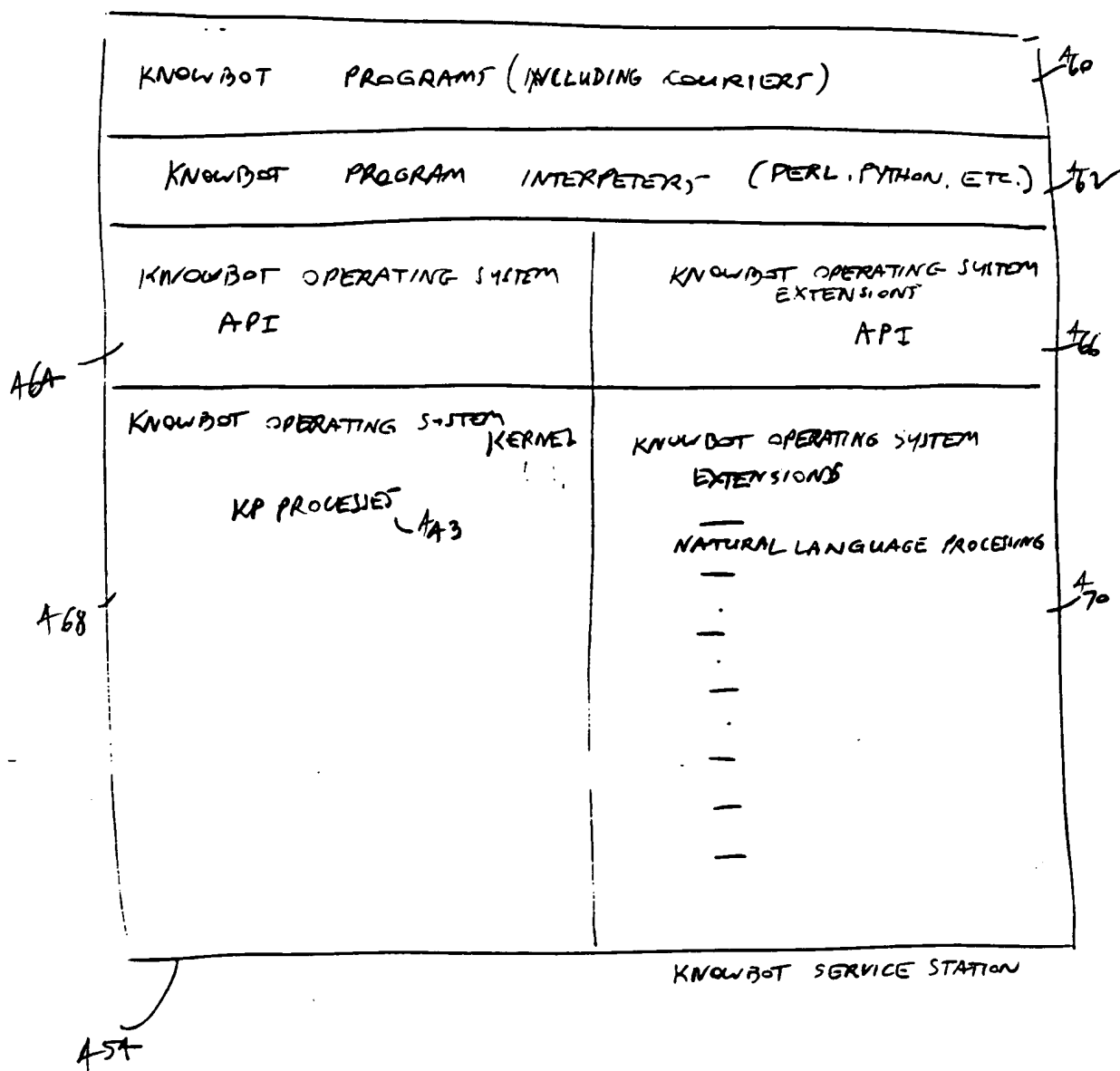


FIGURE 5

ID	AUTH	SCHED	NAV'G'N	TERMS AND CONDITIONS	
110	112	114	116	118	

SYSTEM TIME DATE	OPR'N	PATH	DESC'N OF DATA	DATA
120	122	124	126	128

FIGURE 6

967260-26002780

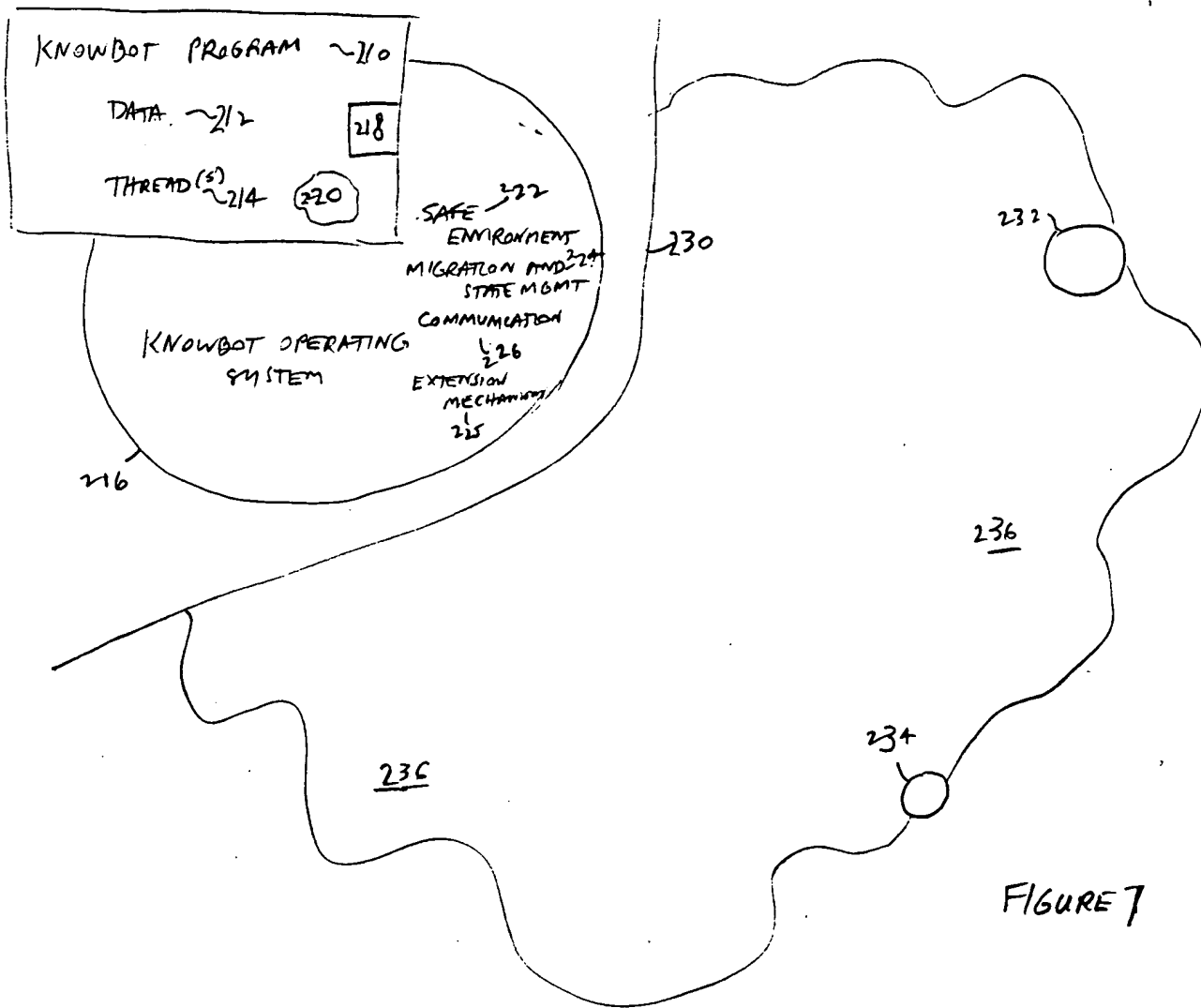


FIGURE 7

96/260-26002/80

CONTAINER ~250

KP SOURCE CODE ~152

CURRENT STATE ~154

SUITCASE ~256

APPLICATION-SPECIFIC DATA ~58

METADATA ~60

KP ORIGIN ~62

NAME OF ENTRY POINT MODULE ~64

EXCEPTION INSTRUCTIONS ~66

OBJECT DATA ~68

REFERENCE TO OBJECTS ~70

FIGURE 8

CLIENT KP ~76

SERVICE REQUEST ~78

CONNECTOR NAME ~80

CONNECTOR INTERFACE TYPE ~82

KOS SUPERVISOR ~242

SURROGATE OBJECT ~284
(CONNECTOR)

CONNECTOR
MANAGER

299

CONNECTOR BROKER ~292

CONNECTOR CLASS OBJECT

294 NAME ~298

TYPE ~300

INSTANCE

296

REGISTER METHOD ~310

UNREGISTER METHOD ~312

LOOKUP METHOD ~314

SERVICE PROVIDING PROCESS ~286

OBJECT RPC MECHANISMS ~288

290

FIGURE 9

KP PROCESSES

KP-210

USER CODE (THREAD) ~214

KOS-216

RPC
CALLS

SUPERVISOR

TRUSTED CODE ~244

RESTRICTED OPERATIONS ~246

BASTION OBJECT ~248

FIGURE 10

REAL OBJECT / 310
 312 — METHODS (UNRESTRICTED)
 :
 :

31A
BASTION OBJECT
METHODS — 315
 316
 ←
 REFERENCES
 TO OBJECT
 OF
 METHODS
 312

FIGURE 11

TOP-LEVEL OBJECT — 340
 — MAIN — (SELF, KOS) — 342
 — SETUP — (SELF, ...) — 344
 :
 — KP — — 346

FIGURE 12

```

import rand                      # Python random number module
import nstools                   # helper module for using KOS namespace

class KP:

    def __init__(self):
        "Initialize KP's instance variables."
        self.maxhops = 20
        self.hopcount = 0
        self.visited = []        # list of KOSes that have been visited

    def __main__(self, kos):
        "Finds services available here, then migrates to a new KOS."
        self.find_services(kos, 'Search.Boolean')
        self.visited.append(kos.get_kos_name())
        self.hopcount = self.hopcount + 1
        if self.hopcount < self.maxhops:
            places = self.get_new_places(kos)
            if places:
                kos.migrate(rand.choice(places))

    def find_services(self, kos, service_type):
        "Save a list of available services in the suitcase"
        services = kos.list_services(service_type)
        file = kos.get_suitcase().open(kos.get_kos_name(), 'w')
        for serv in services:
            file.write(serv.name + '\n')
        file.close()

    def get_new_places(self, kos):
        "Return list of KOSes that have not been visited."
        descriptor = nstools.Lookup(kos.get_namespace(), 'world/kos')
        context = descriptor.Open('Namespace.Context')
        places = []
        for place in context.List():
            if place not in self.visited:
                places.append(place)
        return places

```

FIGURE 13

08/720092-092796

